# C++ ASSIGNMENT

**1.Ques :What is an in-place sorting algorithm?**

a) It needs O(1) or O(logn) memory to create auxiliary locations
b) The input is already sorted and in-place
c) It requires additional storage
d) It requires additional space

**Ans:** a

**2.Ques :In the following scenarios, when will you use selection sort?**

a) The input is already sorted
b) A large file has to be sorted
c) Large values need to be sorted with small keys
d) Small values need to be sorted with large keys

**Ans:** c

**3.Ques :Given an integer array and an integer k where k<=size of array, We need to return the kth smallest element of the array.**

**Ans:**

```
#include<iostream>
#include<vector>
using namespace std;
void insertions(int v[],int n){
    for(int i=1;i<n;i++){
        int j=i;
        while(j>=1&& v[j]<v[j-1]){
            swap(v[j],v[j-1]);
            j--;
        }
```

```
        }
}
int main(){
        int n;
        cout<<"enter the array size :";
        cin>>n;
        int arr[n];
        cout<<"Elements of the array are: ";
        for(int i=0;i<n;i++){
                cin>>arr[i];
        }
        int k;
        cout<<"Enter the kth element: ";
        cin>>k;
        insertions(arr,n);
        cout<<arr[k-1];
return 0;
}
```

**4.Ques:**Given an array, arr[] containing n integers, the task is to find an integer (say K) such that after replacing each and every index of the array by |ai – K| where ( i ∈ [1, n]), results in a sorted array. If no such integer exists that satisfies the above condition then return -1.

**Ans:**

```
#include<iostream>
#include<vector>
#include<climits>
using namespace std;
float max(float a,float b){
        if(a>b) return a;
        else return b;
```

```
}
float min(float a,float b){
	if(a<b) return a;
	else return b;
}
int main(){
	int n;
	cout<<"enter the array size :";
	cin>>n;
	vector<int>arr(n);
	for(int i=0;i<n;i++){
		cin>>arr[i];
	}
	bool flag=true;
	float kmin=(float)INT_MIN;
	float kmax=(float)INT_MAX;
	for(int i=0;i<n-1;i++){
		if(arr[i]>=arr[i+1]){
			kmin=max(kmin,(arr[i]+arr[i+1])/2.0);
		}
		else{
			kmax=min(kmax,(arr[i]+arr[i+1])/2.0);
		}
		if(kmin<kmax){
			flag=false;
			break;
		}
	}
	if(flag==false) cout<<-1;
	else if(kmin==kmax){
```

```
            if(kmin-(int)kmin==0){
                cout<<"There is only value of k: "<<kmin;
            }
            else cout<<-1;
        }
        else{
            if(kmin-(int)kmin>0){
                kmin=(int)kmin+1;
            }
            cout<<"The range is :"<<"["<<kmin<<","<<kmax<<"]";
        }
return 0;
}
```